IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 20, 2018, accepted November 30, 2018, date of publication December 12, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886527

Protograph Based Low-Density Parity-Check Codes Design With Mixed Integer Linear Programming

WOJCIECH SUŁEK, (Member, IEEE)
Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland
e-mail: wojciech.sulek@polsl.pl

The research was supported by the Polish Ministry of Science and Higher Education funding for statutory activities (BK 2018).

ABSTRACT An approach to design protograph-based low-density parity-check (LDPC) codes utilizing mixed integer linear programming (MILP) optimization is presented in this paper. The protograph (base graph) cyclic lifting for a class of quasi-cyclic LDPC codes is considered. In general, the short cycles elimination is the primary optimization goal, possibly weighted by a metric of cycles connectivity. A notion of closed walks in the base graph is shown to be a convenient way for representing sources of cycles in the lifted graph. We express the condition for non-existence of a cycle in the lifted graph corresponding to a closed walk in the base graph in the form of a set of linear inequalities. Such inequalities, collected for all closed walks shorter than a desired limit corresponding to girth, form a set of linear constraints. Meanwhile, the longer closed walks can be reflected in a linear objective function of the optimization. The proposed combination of constraints and objective function forms an input to a MILP solver. As a result, a globally optimized code graph can be obtained. The method can be utilized for binary as well as nonbinary LDPC codes. The numerical results show that the constructed codes can outperform similar codes deigned with reference heuristic search methods.

INDEX TERMS Low density parity check codes, nonbinary codes, protograph, quasi-cyclic codes.

I. INTRODUCTION

Methodical construction of low-density parity check (LDPC) codes [1], [2], binary and nonbinary [3], is one of the major topics for many channel coding researchers, at least for the last decade. While the main objective of the research in this area is to provide possibly the best error correction performance for the specific use-case, the implementation issues typically constraint the design to some class of implementation-oriented LDPC codes.

Currently, most of the LDPC code designs that are considered noteworthy in terms of implementation complexity can be interpreted as long codes derived from short codes by lifting procedure, that is applying copy-and-permutation operations on the protograph [4] (base graph, seed graph). Intense research activities concerning the protograph-based LDPC codes include particularly design of protographs, analysis of the lifting process and development of methods for effective lifting procedure.

Density evolution (DE) and protograph EXIT (PEXIT) techniques are known to facilitate protograph design, typically resulting in the Multi Edge Type (MET) protographs, in which parallel edges are allowed [5]–[10]. MET protograph ensembles have been shown to achieve capacity-approaching thresholds [5], with DE-based optimization enabling to design small-sized protographs, for which the derived LDPC codes are asymptotically close to the capacity for BEC and BIAWGN channels [6]. Numerous other work focus on applications for other channels and specific processing models. In [7], a study on the performance of protograph LDPC codes over Nakagami block-fading channels is presented and a simple design scheme is proposed to construct root-protograph (RP) LDPC codes, designed to cope with block-fading. This research is extended to develop a family of distributed rate-compatible RP codes [8], [9]. In [10], a bandwidth efficient protograph-LDPC based bit-interleaved coded modulation (BICM) is investigated and an approach is proposed that jointly optimizes the protograph node degrees and the mapping of the coded bits to the BICM bit-channels. Also the structured spatially coupled (SC-LDPC) codes [11], [12] are constructed from protographs. In [13] an approach that uses a two-step protograph lifting procedure is proposed, which results in improved minimum distance and

girth properties over a typical one-step lifting. An in-depth survey of the research achievements in the protograph codes design and analysis over a variety of communication systems and channel models can be found in [14].

Extension of protograph codes to the nonbinary domain (over GF($q > 2$)) is also an actively investigated topic, because nonbinary codes can outperform their binary counterparts in the case of short-moderate codeword length and can be seamlessly combined with higher order modulations. Flexible computer search construction methods for nonbinary protograph-based LDPC codes have been presented in several papers [15]–[18]. Nonbinary (2, ν)-regular LDPC codes defined on cages, a special graph structures, which perform very well over high order GF fields, have been shown to be often equivalent to a structured protograph-based codes [19].

When the code design is constrained to cyclic lifting of protographs without parallel edges, the resulting codes, a well known subclass of Quasi-Cyclic LDPC (QC-LDPC) codes [20], [21], are particularly appealing to systems implemented in hardware. The quasi-cyclic feature allows for linear time encoding, while the block-circulant parity check matrix structure enables efficient message routing in a decoder implementation [20], [22]. Parity check matrix of a QC-LDPC code is a grid of submatrices, which are either circulant permutation matrices or all-zero matrices.

Although some special algebraic and geometric structures have been recognized to effectively support QC-LDPC code design [20], [23]–[26], the possible code block lengths and submatrix sizes are typically not flexible in such constructions and the inherent regular degree distribution limits the waterfall performance gain in comparison with irregular distributions. Then a second category of design via the computer search for circulants provide much more flexibility for practical code design.

Commonly the design focus is on finding an optimized permutation of edges in protograph lifting, which for QC-LDPC codes is equivalent to finding cyclic shifts of circulant permutation submatrices in a QC-LDPC parity check matrix. In principle, for a given base matrix and circulant size, it is possible to enumerate all the possible design choices. Then the optimal code can be selected as the one with the greatest coding gain. Such an approach, however, would require an enormous computation efforts, for at least two reasons: 1) the number of enumerated code choices grows exponentially with the number of optimized circulants; 2) LDPC code performance under iterative decoding is typically obtained by Monte Carlo simulations engaging large number of test vectors. To reduce the computational complexity to acceptable level, the first issue can be mitigated by non-exhaustive search utilizing some heuristic algorithm, pseudo-randomly driven design or local optimization, while the second issue is mitigated by an indirect design goal, which is typically elimination of small undesired structures in the code graph, which influence especially the error floor performance. In the case of Belief Propagation (BP) decoding over AWGN channel, these structures are trapping sets [27]–[29], which could be perceived as a set of bits that are likely to reinforce each other in incorrect beliefs. From the Tanner graph perspective, Trapping Sets are nodes arranged in clusters of short cycles. Therefore, an indirect way to remove small trapping sets is to design code graph without short cycles, particularly cycles with low external connectivity [28]–[31].

Due to the computational complexity, an exhaustive computer search of all possible cyclic liftings is possible only for very small graphs, however it has been recently applied for some special cases of short block length codes. In [32] and [33] an exhaustive search methods provided a subclasses of optimized codes, but with relatively small block lengths and fully-connected base graphs only. Other existing research works are based on a non-exhaustive heuristic search, often with pseudo-random component, or optimization leading to a locally optimal outcome. In [34], a hierarchical QC-LDPC code design is proposed, but the cycle elimination is based on a randomly initialized hill climbing technique, which allows for finding a locally optimum solution. In [35], a search loop is proposed that lists all labelings with desired girth among randomly generated choices, then numerical results of performance simulations are used as a code selection criterion. A concept of difference D and double difference DD matrices introduced in [36] facilitates computation of conditions for cycles elimination in cyclic liftings by reducing the number of inequalities that have to be tested. Using difference matrices, some numerical and analytical results have been provided, particularly concerning bounds on the lifting degrees, however these results involve mainly the fully-connected base graphs.

Other research on heuristic search for circulants of QC-LDPC codes, binary and nonbinary, include [37]–[41]. While the existing methods for non-exhaustive exploration of the QC-LDPC code ensembles are miscellaneous, they are mainly based on a kind of loop that either generates random labeling or iteratively adjusts permutation shift values to locally eliminate consecutive cycles. Often, the design goal is only the maximization of the girth. Several works present a design procedures that search for a minimum lifting degree (or minimum code block length) achievable for a desired girth [42], [43]. While this approach can provide a valuable theoretical background, in practice the code design procedure is constrained by the application requirements, which usually includes not only the block length, but also submatrix size, which is equal to the lifting degree. Therefore, in the approach proposed here, the lifting degree is a fixed input parameter and the design algorithm seeks for the best performing code.

Instead of a usual heuristic or pseudo-random search, in this paper a novel approach for cyclic protograph lifting with fixed degree is proposed, which is based on a problem formulation in the form of linear constraints and a global linear objective function. We show how to construct the set of linear conditions that constraint the explored space to solutions ensuring the assumed girth. Moreover, besides the girth conditioning, it is beneficial to additionally eliminate as many girth-length (or even longer) cycles as possible.

For this reason, we show how to formulate the linear objective function that enables maximization of the number of eliminated cycles that has been selected as an optimization goal. In our problem formulation, the set of cycles reflected in the objective function as well as the set of cycles imposing the constraints is freely configurable and left to a specific design choice. Moreover, with every considered cycle, a separate weighting factor is set in the objective function. This factor can reflect a metric of cycle ''harmfulness'', which can be related for example to external connectivity measures like EMD (Extrinsic Message Degree) or ACE (Approximate Cycle EMD).

The linear character of the constraints and objective function makes it possible to search for the globally optimal solution making use of the Mixed Integer Linear Programming (MILP) optimization. We show how to fill the coefficients matrix to express the problem in a vectorized form, by means of a product of coefficient matrix by a vector of optimized variables. This form is ready to use in a numerical solver, because the matrices of coefficients can be sent directly to a typical commercial software implementing MILP.

The proposed code construction method can be used for binary as well as nonbinary designs. In order to obtain a nonbinary code, after graph construction, a nonzero field elements need to be assigned to the edges of the graph. We have employed a method inspired by the binary images selection on a matrix row basis [44]. For the nonbinary construction, the developed algorithm determines GF(q) coefficients associated to the base graph edges, which means a common coefficient is used for every whole submatrix.

In summary, the main contribution of this paper is a new QC-LDPC codes construction algorithm that is based on a novel formulation of the protograph cyclic lifting problem in the form of linear inequality constraints and a linear objective function that reflects the cycles existing in the base graph. It is shown, how this formulation can be vectorized for a MILP solver. The proposed optimization algorithm is rather universal: it can be applied to different classes of QC-LDPC codes, binary and nonbinary, with any regular or irregular single-edge base graphs. Then, in the provided overall QC-LDPC construction algorithm, the waterfall region is optimized by utilizing irregular base graphs with good asymptotic thresholds [15], [45], while the error floor region is optimized by not only the usual girth conditioning, but also global minimization of the short cycles in the lifted graph, possibly weighted by ACE or EMD. To preserve low encoding complexity, the codes with bidiagonal structure can also be constructed. The presented numerical results show performance gains over other recent QC-LDPC and protograph designs for several binary and nonbinary cases.

Although the proposed method is intended mainly for a MILP solver, the separation of the problem formulation by linear equations and the problem solution by numerical solver, enables searching for the solution also with a non-exhaustive, heuristic search. We concentrate on formulation of the code design problem in the aforementioned form, but refrain from investigating the solving algorithm as such. In the numerical study an existing MILP software is utilized.

In section II, we introduce notations, definitions of QC-LDPC code class and MILP optimization as well as discuss some basic code graph properties. The main ideas are provided in section III, where we formulate the code design problem, express it in the form convenient for MILP, propose a simple practical code design strategies, including the nonbinary codes construction case and finally summarize the overall QC-LDPC construction algorithm. In section IV we study several code design cases and provide numerical results, then in section V we summarize the research.

II. PRELIMINARIES

A. MIXED INTEGER LINEAR PROGRAMMING

The constrained linear optimization problem, with integer restrictions on some of the variables, is known as Mixed Integer Linear Programming and can be expressed as follows: find integer values of $y_1, y_2, \ldots, y_n$ and nonnegative real values of $x_1, x_2, \ldots, x_p$ that minimize the objective function:

$$z = \sum_{j=1}^{n} c_j y_j + \sum_{k=n+1}^{p} c_k x_k \tag{1}$$

$$\text{subject to: } \sum_{j=1}^{n} a_{ij} y_j + \sum_{k=n+1}^{p} a_{ik} x_k \le b_i \quad (i = 1, 2, \ldots, m) \tag{2}$$

$$y_j = 0, 1, 2, \ldots \quad (j = 1, 2, \ldots, n) \tag{3}$$

$$x_k \ge 0 \quad (k = n+1, \ldots, p) \tag{4}$$

where c_j, c_k are coefficients defining the objective function in (1), while a_{ij}, a_{ik} and b_i are coefficients defining the m problem constraints in (2).

For this research, the optimization variables are shift values in cyclic protograph lifting, which are nonnegative integers. Therefore the pure Integer Programming (IP) is applied, which means the problem formulation includes only components with integer variables $y_1, y_2, \ldots, y_n$ in (1)-(4) and the MILP problem is defined by coefficients c_j, a_{ij} and b_j for $i = 1, 2, \ldots, m$, and $j = 1, 2, \ldots, n$.

B. QC-LDPC CODE PARITY CHECK MATRIX

A binary QC-LDPC code [20] is characterized by parity check matrix that is an array of circulants. Commonly used circulants are square submatrices, which are either all-zero matrices or circulant permutations of an identity matrix.

Let $\mathbf{P}^s$ be a circulant of size $P \times P$ obtained by cyclic shifting the identity matrix $\mathbf{I}_{P\times P}$ by s positions to the right. In this research, the class of codes with the following structure of the parity check matrix is considered:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^{s_{1,1}} & \mathbf{P}^{s_{1,2}} & \ldots & \mathbf{P}^{s_{1,J}} \\ \mathbf{P}^{s_{2,1}} & \mathbf{P}^{s_{2,2}} & \ldots & \mathbf{P}^{s_{2,J}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{s_{I,1}} & \mathbf{P}^{s_{I,2}} & \ldots & \mathbf{P}^{s_{I,J}} \end{bmatrix} \tag{5}$$

where $s_{i,j} \in \{0, 1, \ldots, P-1, \infty\}$, where $\mathbf{P}^{\infty}$ is used to denote all-zero matrix [21].

Codes associated with parity check matrix (5) are (N, K) QC-LDPC codes with $N = JP$, $K = (J - I)P$ and rate $R = 1 - I/J$, when $\mathbf{H}$ has full rank.

C. BASE GRAPH CYCLIC LIFTING

The protograph of a subclass of QC-LDPC codes defined in (5) belongs to a narrowed group of protographs with no parallel edges, for which we use the term of base graph throughout this paper. The base graph corresponds to the base matrix, which we denote $\mathbf{W}_{I\times J}$. Base matrix can be extracted from parity check matrix (5) by replacing zero matrices and circulant matrices in $\mathbf{H}$ with 0 and 1, respectively.

Let us denote the base graph by $\mathcal{G} = (\mathcal{V}_v \cup \mathcal{V}_c, \mathcal{E})$, which is a bipartite graph composed of $\mathcal{V}_v = \{v_1, v_2, \ldots, v_J\}$ – the set of variable nodes, $\mathcal{V}_c = \{c_1, c_2, \ldots, c_I\}$ – the set of check nodes, and $\mathcal{E} \subseteq \mathcal{V}_v \times \mathcal{V}_c$ – the set of edges, $\mathcal{E} = \{e_1, e_2, \ldots, e_T\}$. An edge $e_t = (v_j, c_i)$ belongs to $\mathcal{E}$ if and only if $w_{ij} \neq 0$ in the base matrix $\mathbf{W}$. Conversely, the base matrix $\mathbf{W}$ is a biadjacency matrix of the base graph $\mathcal{G}$. The total number of edges in $\mathcal{G}$, denoted by T, is equal to the number of nonzero elements in the base matrix $\mathbf{W}$.

Let $\mathcal{P} = \{1, 2, \ldots, p, \ldots, P\}$ be a finite set, where P is a graph lifting degree. The lifted graph, which is a QC-LDPC code graph, has vertex sets $V_c \times \mathcal{P}$ and $V_v \times \mathcal{P}$. For each $e_t \in \mathcal{E}$, a circulant permutation is assigned, characterized by a permutation shift s_t, that is: $\theta_{e_t}(p) \equiv p + s_t \mod P$. Then, the edge set is $\mathcal{E} \times \mathcal{P}$, where an edge $(e_t, p) = (v_j, c_i, p) \in \mathcal{E} \times \mathcal{P}$ is incident with vertices (v_j, p) and $(c_i, \theta_{e_t}(p))$. The cyclic lifting process is thus characterized by the set of permutation shift values $\mathcal{S} = \{s_1, s_2, \ldots s_T\}$, $0 \leq s_t < P$. The lifted graph $\mathcal{G}^{(\mathcal{P},\mathcal{S})} = ((\mathcal{V}_v \times \mathcal{P}) \cup (\mathcal{V}_c \times \mathcal{P}),\ \mathcal{E} \times \mathcal{P})$ is a bipartite graph of a QC-LDPC code, with PJ variable nodes, PI check nodes and PT edges.

Here and in the following sections, for simple notation, a single-indexed s_t is used to denote a permutation shift value in $\mathbf{H}$. The elements of $\mathcal{S} = \{s_1, s_2, \ldots s_T\}$ correspond to T nonzero submatrices $\mathbf{P}^{s_{i,j}}$ in (5). It means that every $s_{i,j} \neq \infty$ in (5) is denoted with one of the single-indexed $s_t \in \mathcal{S}$, assigned to the edge $e_t \in \mathcal{E}$.

Any code from the considered QC-LDPC subclass is fully described by its base matrix $\mathbf{W}$, lifting degree P and the set of permutation shift values $\mathcal{S}$.

D. CYCLES IN THE CODE GRAPH

It has been well recognized that short cycles in the code graph significantly degrade performance of the corresponding LDPC code. A cycle of length $2l$ can be perceived as a sequence of adjacent edges, which starts and ends at the same vertex, and satisfies the condition that no edge appears more than once in the sequence. Let this closed sequence of edges be denoted as $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$, where $t_1, t_2, \ldots, t_{2l}$ indicate edges in the cycle.

Two subsequent edges in any cycle correspond to distinct circulant permutation matrices, which are either in the same row block, or in the same column block [20]. Hence, any cycle of length $2l$ in a QC-LDPC code graph can be represented by the ordered series of nonzero circulant submatrices:

$$\mathbf{P}^{s_{t_1}} \rightarrow \mathbf{P}^{s_{t_2}} \rightarrow \cdots \rightarrow \mathbf{P}^{s_{t_{2l}}} \rightarrow \mathbf{P}^{s_{t_1}} \tag{6}$$

where $t_i \neq t_{i+1}$ for $1 \leq i < 2l$ and any subsequent $\mathbf{P}^{s_{t_i}}, \mathbf{P}^{s_{t_{i+1}}}$ are located in either the same column block or the same row block of $\mathbf{H}$, while any $\mathbf{P}^{s_{t_i}}$ and $\mathbf{P}^{s_{t_{i+2}}}$ are located in distinct column blocks and row blocks [21]. However, in general the chain (6) can contain submatrices that reappear in positions $i + 4$ or greater. Therefore, the corresponding sequence of edges in the base graph $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$ need not be strictly a cycle. It is a closed sequence of adjacent edges in the graph, without restriction of their single appearance. For such a sequence we use a term of closed walk [46].

Every closed walk existing in the base graph corresponds to a chain of circulants (6). However, not every closed chain is necessarily a source of cycles in the lifted graph.

The ordered series of permutation shift values:

$$s_{t_1};\ s_{t_2};\ s_{t_3};\ \cdots s_{t_{2l}} \tag{7}$$

can be used to specify the condition for existence of corresponding cycles in the cyclically lifted graph. It has been shown [20], [21] that a lifted graph contains cycles corresponding to a closed walk $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$ if and only if $\sum_{k=1}^{l} \Delta_{t_{2k}} \equiv 0 \mod P$, where $\Delta_{t_{2k}} = s_{t_{2k-1}} - s_{t_{2k}}$. Below we reformulate this condition with a following theorem characterizing the relationship between cycles in the lifted graph and corresponding closed walks in the base graph.

Theorem 1: The cyclically lifted graph $\mathcal{G}^{(\mathcal{P},\mathcal{S})}$ contains P length-$2l$ cycles $(e_{t_1}, p) \sim (e_{t_2}, p) \sim \cdots \sim (e_{t_{2l}}, p) \sim$ corresponding to a length-$2l$ closed walk $e_{t_1} \sim e_{t_2} \sim \cdots \sim e_{t_{2l}} \sim$ in the base graph $\mathcal{G}$, if and only if the following condition is satisfied for the permutation shift values $s_{t_1}, s_{t_2}, \ldots, s_{t_{2l}}$ assigned to the edges of the closed walk:

$$\sum_{k=1}^{2l} (-1)^{k-1} s_{t_k} \equiv 0 \mod P \tag{8}$$

E. EXTERNAL CONNECTIVITY OF CYCLES

Cycles do not contribute to the error rate uniformly and construction methods taking into consideration external connectivity of cycles have demonstrated to improve results. Two common measures used to calculate the connectivity of cycles are Extrinsic Message Degree (EMD) and Approximate Cycle EMD (ACE). The EMD of a cycle is the number of check nodes that are singly connected to the variable node subset forming this cycle [28]. The ACE of a length $2l$ cycle is $\sum_i (d_i - 2)$, where d_i is the degree of the ith variable node in this cycle [28]. If there are no variable nodes in a cycle that share common check nodes outside of the cycle, then the EMD of this cycle is equal to the ACE. Similar definitions of EMD and ACE can be stated for closed walks.

Performance of an LDPC code with iterative BP decoding is strictly connected to its graph structure.

Particularly, in [27] Richardson showed that error floors in BP decoders are generally caused by trapping sets. A trapping set is a set of a small number of bits that reinforce each other in their incorrect beliefs. Trapping sets of bits are arranged by clusters of short cycles in a corresponding Tanner graph. Therefore, one way to try to remove trapping sets is to design the Tanner graph carefully so that the dangerous clusters of short cycles do not exist [29].

This can be achieved by avoiding short cycles with low external connectivity, that is cycles with low EMD (or ACE) parameters. Cycles having small EMD are more prone to induce small trapping sets [28]. Minimization of short cycles with low EMD is the basis of many published code construction methods, e.g. [28], [30], [31].

III. THE PROPOSED OPTIMIZATION METHOD

In a common practical LDPC system design strategy, the base matrix $\mathbf{W}$ and lifting degree P are imposed by the desired variable node degree distribution on one hand, and the implementation details on the other hand, such as the decoder architecture and required throughput. Then, the crucial stage for optimization of a graph properties in a QC-LDPC code is an effective search for permutation shift values. The goal is to find values $s_t \in \mathcal{S}$, such that congruencies of type (8) are avoided for short closed walks in the base graph.

Therefore, the first step in the proposed graph optimization method is the base graph analysis in order to find all closed walks of length up to an assumed limit $2l_{\max}$. To achieve this, for every variable node, a tree is expanded up to the depth $l_{\max}$, according to the base graph structure. For every node that appears in the tree more than once, a closed walk is identified by backtracking the tree.

Let denote the total number of closed walks detected in the base graph by D and let assign an index d to every closed walk, where $d = 1, 2, \ldots, D$. Let dth closed walk be of length $L(d)$ and include edges of indexes $t_{1,d}, t_{2,d}, \ldots, t_{L(d),d}$, that is $e_{t_{1,d}}, e_{t_{2,d}}, \ldots, e_{t_{L(d),d}}$. According to (8), condition for non-existence of cycles in the covering graph, resulting from dth closed walk in the base graph, is:

$$\sum_{k=1}^{L(d)} (-1)^{k-1} s_{t_{k,d}} \not\equiv 0 \mod P \tag{9}$$

In order to express a series of such conditions in a matrix notation, the sum in (9) need to be formulated over all optimized variables in $\{s_1, s_2, \ldots s_T\}$, that is in the form $\sum_{t=1}^{T} a_t s_t$. Here, every coefficient a_t $(t = 1, \ldots, T)$ can be perceived as an indication of how many times and in which direction an edge e_t is passed in the considered closed walk. To illustrate this, we use the convention with a directed base graph, where edge directions are from variable nodes to check nodes. Example is shown in Fig. 1(a), where a small graph is shown that contains edges $e_1, \ldots, e_9$, hence every designated closed walk is defined by coefficients $a_1, \ldots, a_9$.

When tracking a closed walk in such a directed graph, for every edge e_t passed in a forward direction (consistent with an edge direction), the corresponding coefficient a_t is increased by 1. Likewise, for every edge e_t passed in a backward direction, the corresponding coefficient is decreased by 1. This is illustrated in Figs. 1(b)-(d). For example, the length-8 closed walk $e_8 \sim e_9 \sim e_5 \sim e_3 \sim e_8 \sim e_9 \sim e_5 \sim e_3 \sim$ shown in Fig. 1(c) corresponds to coefficients: $a_8 = 2$, $a_9 = -2$, $a_5 = 2$, $a_3 = -2$, and $a_1 = a_2 = a_4 = a_6 = a_7 = 0$.

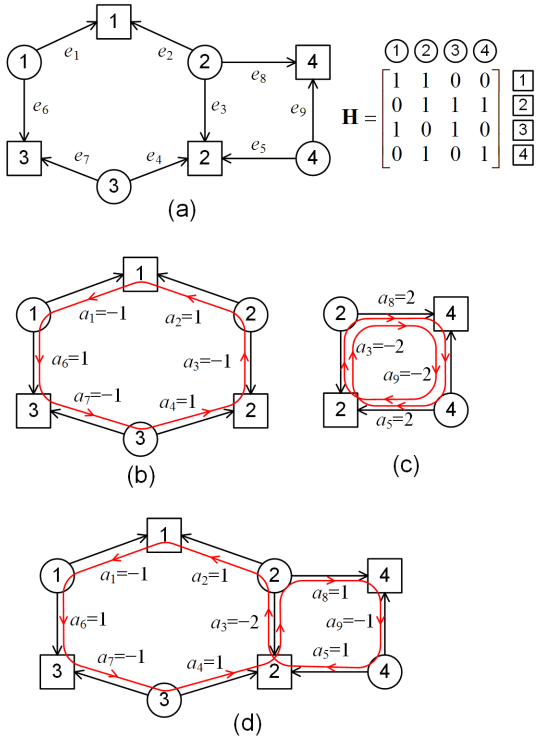

FIGURE 1. An example graph (a) and its closed walks of length-6 (b), length-8 (c) and length-10 (d). Variable nodes are illustrated by circles and check nodes are illustrated by squares.

Let the coefficients a_t, $t = 1, \ldots, T$ for the dth closed walk be denoted as $a_t(d)$, $d = 1, \ldots, D$. Arranging coefficients $a_t(d)$ in a vector $\mathbf{a}(d) = [a_1(d), a_2(d), \ldots, a_T(d)]$ and variables s_t in a vector $\mathbf{s} = [s_1, s_2, \ldots, s_T]$ allows to reformulate the condition (9) with a compact vector by vector multiplication form, which for dth closed walk in the base graph is as follows:

$$\sum_{t=1}^{T} a_t(d) s_t = \mathbf{a}(d) \cdot \mathbf{s}^T \not\equiv 0 \mod P \tag{10}$$

where $a_t(d)$ is the difference between the number of passes through the edge e_t in forward and backward directions in dth closed walk. If the edge e_t is not passed at all in dth closed walk, then $a_t(d) = 0$. Similar vectorized formulation like in (10) was previously proposed in [35].

For every closed walk that should be eliminated in a lifting process, a condition like in (10) could possibly entail a constraint in an IP optimization problem. However, this requires reformulating the mod P incongruence in the form of linear inequalities as in (2). At first we will demonstrate how to do this for the simplest case of a length-4 cycle.

A. LINEAR CONSTRAINTS FOR THE LENGTH-4 CLOSED WALK CASE

For a length-4 closed walk, the product $\mathbf{a}(d) \cdot \mathbf{s}^T$ includes 4 nonzero components $a_t s_t$, where $a_t = +1$ for two of them and $a_t = -1$ for another two. Then, since $0 \leq s_t < P$, the upper and lower bound on the value of $\mathbf{a}(d) \cdot \mathbf{s}^T$ is easily determined to be: $-2(P-1) \leq \mathbf{a}(d) \cdot \mathbf{s}^T \leq 2(P-1)$. Such a bounded value is $\bmod P$ incongruent to 0, if it is different from $-P$, 0 and P. It is equivalent to fulfilling one of the following conditions, corresponding to 4 possible disjoint ranges:

$$\begin{aligned} -P < \mathbf{a}(d) \cdot \mathbf{s}^T < 0 \\ \vee 0 < \mathbf{a}(d) \cdot \mathbf{s}^T < P \\ \vee \mathbf{a}(d) \cdot \mathbf{s}^T < -P \\ \vee \mathbf{a}(d) \cdot \mathbf{s}^T > P \end{aligned} \tag{11}$$

Since the sum in $\mathbf{a}(d) \cdot \mathbf{s}^T$ contains only integer components, we can rewrite (11) in a form utilizing less-or-equal operators:

$$\begin{aligned} &\left(\mathbf{a}(d) \cdot \mathbf{s}^T \leq -1 \wedge -\mathbf{a}(d) \cdot \mathbf{s}^T \leq P-1\right) \\ &\vee \left(-\mathbf{a}(d) \cdot \mathbf{s}^T \leq -1 \wedge \mathbf{a}(d) \cdot \mathbf{s}^T \leq P-1\right) \\ &\vee \left(\mathbf{a}(d) \cdot \mathbf{s}^T \leq -P-1\right) \\ &\vee \left(-\mathbf{a}(d) \cdot \mathbf{s}^T \leq -P-1\right) \end{aligned} \tag{12}$$

Among the four alternative conditions in (12), only one could hold at a time, thus further reformulation is necessary to obtain a series of inequalities that should be fulfilled jointly, as in the IP problem formulation (2). Let $y_1, \ldots, y_4$ be additional binary (0-1) variables. In every of the 4 conditions, marked by $i = 1, \ldots, 4$, we subtract a component Zy_i from the left-hand side of inequality, where Z is a positive constant factor, large enough to assure the inequality always holds for $y_i = 1$, regardless of the remaining variables. If as a result of optimization process y_i is 0, the ith alternative condition is satisfied for IP optimization results in $\mathbf{s}$. On the other hand, $y_i = 1$ means the ith condition is actually not constraining the values in $\mathbf{s}$, so it can be perceived redundant (as a result of optimization). Then, the whole set of inequalities that should jointly hold to obtain the $\bmod P$ incongruence in (10) is:

$$\begin{aligned} \mathbf{a}(d) \cdot \mathbf{s}^T - Zy_1 &\leq -1 \\ -\mathbf{a}(d) \cdot \mathbf{s}^T - Zy_1 &\leq P-1 \\ -\mathbf{a}(d) \cdot \mathbf{s}^T - Zy_2 &\leq -1 \\ \mathbf{a}(d) \cdot \mathbf{s}^T - Zy_2 &\leq P-1 \\ \mathbf{a}(d) \cdot \mathbf{s}^T - Zy_3 &\leq -P-1 \\ -\mathbf{a}(d) \cdot \mathbf{s}^T - Zy_4 &\leq -P-1 \\ y_1 + y_2 + y_3 + y_4 &\leq 3 \end{aligned} \tag{13}$$

where the last inequality ensures that one of the binary variables $y_1, \ldots, y_4$ takes the value of 0, therefore one of the alternative conditions holds.

With constraints defined as in (13), any proper solution found by numerical solver guarantees to eliminate corresponding cycles from the code graph. However, too many constraints can make the optimization problem unsolvable, and it is usually not known in advance: how many cycles can be eliminated. Then, a desirable possibility is to convert some of the conditions to contribute to the optimization goal function instead of strictly constraining the design. To accomplish this, the last inequality in (13) is dropped from problem constraints, while the optimization goal function includes a component:

$$z = \ldots + \alpha(d)(y_1 + y_2 + y_3 + y_4) + \ldots \tag{14}$$

where $\alpha(d)$ is a weighting factor for dth closed walk. Then, the optimization solution can include both eliminated cycles (for which $\sum y_i = 3$) and not eliminated cycles (for which $\sum y_i = 4$), with maximized number of eliminated cycles, if $\alpha(d)$ is constant over d.

In a general case, the weight $\alpha(d)$ enables assigning level of harmfulness to every closed walks separately. It can be related to the closed walk length or other characteristics like an EMD (Extrinsic Message Degree) [28] of corresponding cycles, if a more complex optimization strategy is desired.

In order to prepare data for the numerical solver, it is convenient to express the set of constraints (13) in a matrix form:

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{s}^T \\ \mathbf{y}^T \end{bmatrix} \leq \mathbf{b}^T \tag{15}$$

where $\mathbf{y}$ is a vector with variables $y_1, y_2, \ldots$, while $\mathbf{A}$ and $\mathbf{b}$ contain coefficients of the set of inequalities. For example, the rows of $\mathbf{A}$ and $\mathbf{b}$ corresponding to the first ($d = 1$) cycle of length-4, collected in submatrices denoted as $\mathbf{A}(1)$ and $\mathbf{b}(1)$, according to (13) are:

$$\mathbf{A}(1) = \begin{bmatrix} \mathbf{a}(1) & -Z & 0 & 0 & 0 & 0 & \\ -\mathbf{a}(1) & -Z & 0 & 0 & 0 & 0 & \\ -\mathbf{a}(1) & 0 & -Z & 0 & 0 & 0 & \\ \mathbf{a}(1) & 0 & -Z & 0 & 0 & 0 & \cdots \\ \mathbf{a}(1) & 0 & 0 & -Z & 0 & 0 & \\ -\mathbf{a}(1) & 0 & 0 & 0 & -Z & 0 & \\ \mathbf{0} & 1 & 1 & 1 & 1 & 0 & \end{bmatrix} \tag{16}$$

$$\mathbf{b}(1)^T = \begin{bmatrix} -1 \\ P-1 \\ -1 \\ P-1 \\ -P-1 \\ -P-1 \\ 3 \end{bmatrix} \tag{17}$$

Similarly the subsequent components $\mathbf{A}(1), \mathbf{A}(2), \ldots, \mathbf{A}(D)$ and $\mathbf{b}(1), \mathbf{b}(2), \ldots, \mathbf{b}(D)$ are created. They are assembled column-wise to form the whole constraint coefficient matrices $\mathbf{A}$ and $\mathbf{b}$.

B. GENERAL IP FORMULATION

The QC-LDPC code graph optimization problem can be expressed in a vectorized form as follows: compute the elements of integer vectors $\mathbf{s} = [s_1, s_2, \ldots, s_T]$, $0 \le s_t < P$ and $\mathbf{y} = [y_1, y_2, \ldots, y_U]$, $0 \le y_u \le 1$ that minimize:

$$z = \left[\mathbf{0}_{1\times T},\ \mathbf{c}\right] \cdot \begin{bmatrix} \mathbf{s}^T \\ \mathbf{y}^T \end{bmatrix} \tag{18}$$

subject to:

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{s}^T \\ \mathbf{y}^T \end{bmatrix} \le \mathbf{b}^T \tag{19}$$

where the coefficient matrix $\mathbf{A}$ is composed of submatrices $\mathbf{A}(d)$ arranged column-wise:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}(1) \\ \mathbf{A}(2) \\ \vdots \\ \mathbf{A}(D) \end{bmatrix} = \mathrm{col}[\mathbf{A}(1), \mathbf{A}(2), \cdots, \mathbf{A}(D)] \tag{20}$$

corresponding to the list of D closed walks, and similarly vectors $\mathbf{b}$ and $\mathbf{c}$ are composed of subvectors $\mathbf{b} = \mathrm{col}[\mathbf{b}(1), \mathbf{b}(2), \cdots, \mathbf{b}(D)]$ and $\mathbf{c} = [\mathbf{c}(1), \mathbf{c}(2), \cdots, \mathbf{c}(D)]$ corresponding to D closed walks.

Contents of the component coefficient submatrices can be defined by generalizing results from the previous subsection to the length-$L(d)$ cycle. The value of $\mathbf{a}(d) \cdot \mathbf{s}^T$ is bounded in general by: $-\frac{L(d)}{2}(P-1) \le \mathbf{a}(d) \cdot \mathbf{s}^T \le \frac{L(d)}{2}(P-1)$, so every additional 2 edges in the considered closed walk entails 2 more disjoint ranges, defined by 4 inequalities expressing the incongruence condition and employing 2 more variables y_u. Coefficient submatrices $\mathbf{A}(d)$, $\mathbf{b}(d)$ and $\mathbf{c}(d)$ can be expressed as follows.

- In the case of strict constraint, that is if the optimization solution should guarantee to eliminate cycles corresponding to the dth closed walk, $\mathbf{A}(d)$ and $\mathbf{b}(d)^T$ contains $2L(d)-1$ rows as shown in (23), while $\mathbf{c}(d) = [0, \ldots, 0]_{1\times L(d)}$ (all-zero vector of length $L(d)$).
- If the optimization solution need not guarantee to eliminate cycles, but the optimization goal should be greater by $\alpha(d)$ for optimization results with not eliminated cycles corresponding to the dth closed walk, $\mathbf{A}(d)$ and $\mathbf{b}(d)^T$ contains the first $2L(d)-2$ rows of matrices shown in (23) (all rows except for the last row imposing the strict constraint), while $c(d)$ is composed of $\alpha(d)$:

$$\mathbf{c}(d) = [\alpha(d), \alpha(d), \ldots, \alpha(d)]_{1\times L(d)} \tag{21}$$

Optimization variable vectors $\mathbf{s}$ and $\mathbf{y}$ have T and $U = \sum_{d=1}^{D} L(d)$ elements respectively, hence the number of columns in $\mathbf{A}$ is $T+U$. Meanwhile, the number of rows in $\mathbf{A}$ and $\mathbf{b}$ depends on the number of considered closed walks.

Setting the problem in the presented matrix form facilitate utilization of any optimization software supporting Integer Programming. For example, the coefficient matrices $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, as they are defined in this article, can be used directly in `intlinprog` function of Matlab environment.

C. EXAMPLE

As an example let us consider a cyclic lifting of a small graph shown in Fig. 1(a). The lifted graph corresponds to a matrix with 9 nonzero circulants:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^{s_1} & \mathbf{P}^{s_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^{s_3} & \mathbf{P}^{s_4} & \mathbf{P}^{s_5} \\ \mathbf{P}^{s_6} & \mathbf{0} & \mathbf{P}^{s_7} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^{s_8} & \mathbf{0} & \mathbf{P}^{s_9} \end{bmatrix} \tag{22}$$

therefore the vector of cyclic shift values is $\mathbf{s} = [s_1, s_2, \ldots, s_9]$. The proposed procedure can be used to achieve a lifted graph with an assumed girth g. For example, if $g = 8$ is required, a strict constraints for all length-4 and length-6 closed walks should be formulated. The base graph contains one length-4 closed walk $e_3 \sim e_5 \sim e_9 \sim e_8 \sim$, for which the condition for non-existence of cycles in the lifted graph is $s_3 - s_5 + s_9 - s_8 \not\equiv 0 \mod P$. This condition, formulated as in (13) requires additional binary variables $\mathbf{y} = [y_1, y_2, y_3, y_4]$. Corresponding matrices of coefficients $\mathbf{A}(1)$ and $\mathbf{b}(1)$ for lifting degree $P = 9$, constructed according to (16)-(17) or (23), are presented in Fig. 2. The constant Z is set to $Z = 3P = 27$, which is a sufficiently large value for a length-4 cycle case.

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	
0	0	1	0	-1	0	0	-1	1	-27	0	0	0	0	0	0	0	0	0	-1
0	0	-1	0	1	0	0	1	-1	-27	0	0	0	0	0	0	0	0	0	8
0	0	-1	0	1	0	0	1	-1	0	-27	0	0	0	0	0	0	0	0	-1
0	0	1	0	-1	0	0	-1	1	0	-27	0	0	0	0	0	0	0	0	8
0	0	1	0	-1	0	0	-1	1	0	0	-27	0	0	0	0	0	0	0	-10
0	0	-1	0	1	0	0	1	-1	0	0	0	-27	0	0	0	0	0	0	-10
0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	3

$\mathbf{A}(1)$ $\mathbf{b}(1)^T$

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	
1	-1	1	-1	0	-1	1	0	0	0	0	0	0	-45	0	0	0	0	0	-1
-1	1	-1	1	0	1	-1	0	0	0	0	0	0	-45	0	0	0	0	0	8
1	-1	1	-1	0	-1	1	0	0	0	0	0	0	0	-45	0	0	0	0	-10
-1	1	-1	1	0	1	-1	0	0	0	0	0	0	0	-45	0	0	0	0	17
-1	1	-1	1	0	1	-1	0	0	0	0	0	0	0	0	-45	0	0	0	-1
1	-1	1	-1	0	-1	1	0	0	0	0	0	0	0	0	-45	0	0	0	8
-1	1	-1	1	0	1	-1	0	0	0	0	0	0	0	0	0	-45	0	0	-10
1	-1	1	-1	0	-1	1	0	0	0	0	0	0	0	0	0	0	-45	0	17
1	-1	1	-1	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	-45	-19
-1	1	-1	1	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	-45	-19
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	5

$\mathbf{A}(2)$ $\mathbf{b}(2)^T$

FIGURE 2. Matrices of coefficients defining linear constraints for elimination of the length-4 cycle (A(1) and b(1)) and the length-6 cycle (A(2) and b(2)) by lifting graph presented in Fig. 1 with $P = 9$.

The base graph also contains one length-6 closed walk $e_1 \sim e_2 \sim e_3 \sim e_4 \sim e_7 \sim e_6 \sim$, for which the condition is $s_1 - s_2 + s_3 - s_4 + s_7 - s_6 \not\equiv 0 \mod P$ and the linear formulation requires additional binary variables $\mathbf{y} = [y_5, y_6, \ldots, y_{10}]$. Vectorized formulation, according to (23), involves matrices $\mathbf{A}(2)$ and $\mathbf{b}(2)$ shown in the bottom of Fig. 2. In this case, $Z = 5P = 45$, which is a sufficiently large value for a length-6 cycle case.

To obtain a girth-8 graph, the MILP procedure should seek for any solution satisfying constraints (19), with $\mathbf{A} = \mathrm{col}[\mathbf{A}(1), \mathbf{A}(2)]$, $\mathbf{b} = [\mathbf{b}(1), \mathbf{b}(2)]$, $P = 9$ and the goal function coefficient vector in (18) set to zero, that is $\mathbf{c} = \mathbf{0}$.

$$\mathbf{A}(d)=\left[\begin{array}{ccccccccccccc}
\mathbf{a}(d) & 0 & & 0 & -Z & & 0 & & 0 & 0 & 0 & & 0 \\
-\mathbf{a}(d) & 0 & & 0 & -Z & & 0 & & 0 & 0 & 0 & & 0 \\
\vdots & & & & & \ddots & & & & & \vdots & & \\
\hline
-\mathbf{a}(d) & 0 & & 0 & 0 & & -Z & & 0 & 0 & 0 & & 0 \\
\mathbf{a}(d) & 0 & \cdots & 0 & 0 & & -Z & & 0 & 0 & 0 & \cdots & 0 \\
\vdots & & & & & & & \ddots & & & \vdots & & \\
\hline
\mathbf{a}(d) & 0 & & 0 & 0 & & 0 & & -Z & 0 & 0 & & 0 \\
-\mathbf{a}(d) & 0 & & 0 & 0 & & 0 & & 0 & -Z & 0 & & 0 \\
\hline
\mathbf{0} & 0 & & 0 & 1 & \cdots & 1 & \cdots & 1 & 1 & 0 & & 0
\end{array}\right], \quad \mathbf{b}(d)^T=\left[\begin{array}{c}
-1 \\ P-1 \\ -P-1 \\ 2P-1 \\ \vdots \\ (\frac{L(d)}{2}-1)P-1 \\ \hline -1 \\ P-1 \\ -P-1 \\ 2P-1 \\ \vdots \\ (\frac{L(d)}{2}-1)P-1 \\ \hline -(\frac{L(d)}{2}-1)P-1 \\ -(\frac{L(d)}{2}-1)P-1 \\ \hline L(d)-1
\end{array}\right] \tag{23}$$

As a result of MILP optimization procedure, a following vectors are obtained: $\mathbf{s} = [0, 8, 0, 8, 0, 8, 0, 1, 0]$ and $\mathbf{y} = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1]$. In this result, the value of $y_1 = 0$ ensures fulfillment of inequality $y_1 + y_2 + y_3 + y_4 \leq 3$ defined by the last row of $\mathbf{A}(1)$ and last element of $\mathbf{b}(1)$. Constraints corresponding to the first two rows of $\mathbf{A}(1)$, for the $\mathbf{s}$, $\mathbf{y}$ listed above (where $y_1 = 0$) can be verified as $0 - 0 - 1 + 0 \leq -1$ and $-0 + 0 + 1 - 0 \leq 8$ respectively. Meanwhile, constraints defined by rows $3 \ldots 6$ of $\mathbf{A}(1)$ hold because of the dominating components with coefficient $Z = -27$ multiplying variables $y_2 = y_3 = y_4 = 1$. It means the original constraint $s_3 - s_5 + s_9 - s_8 \not\equiv 0 \mod P$ holds, particularly $-P < s_3 - s_5 + s_9 - s_8 < 0$.

Similarly, for the length-6 closed walk, since $y_9 = 0$ and $y_5 = y_6 = y_7 = y_8 = y_{10} = 1$, the constraint involving y_9 can be verified as $0 - 8 + 0 - 8 - 8 + 0 \leq -19$ (where $-19 = -2P - 1$), while the remaining constraints hold due to the dominating components with coefficient $Z = -45$. It means the original constraint $s_1 - s_2 + s_3 - s_4 + s_7 - s_6 \not\equiv 0 \mod P$ holds, particularly $s_1 - s_2 + s_3 - s_4 + s_7 - s_6 < -2P$.

As a results, the matrix as in (22), with lifting degree $P = 9$ and the obtained $\mathbf{s} = [0, 8, 0, 8, 0, 8, 0, 1, 0]$ is associated with a girth-8 bipartite graph. Further studies reveal that the example small graph contains 2 length-8 closed walks, 1 length-10 closed walks, 3 length-12, 3 length-14 closed walks and 5 length-16 closed walks. Matrices $\mathbf{A}(3) \ldots \mathbf{A}(16)$, $\mathbf{b}(3) \ldots \mathbf{b}(16)$ for these closed walks of length-8 and longer can be formulated similarly to the presented in Fig. 2. Then, if the last rows from $\mathbf{A}(3) \ldots \mathbf{A}(16)$, $\mathbf{b}(3) \ldots \mathbf{b}(16)$ are excluded, while the corresponding elements in $\mathbf{c}$ are set to one, the graph lifting can be optimized to eliminate the maximum number of these longer cycles. It turns out that for $P = 9$, the MILP optimization procedure gives as a result $\mathbf{s} = [8, 0, 2, 0, 8, 0, 5, 8, 0]$, for which all those cycles are avoided. Therefore, the associated graph is a girth-18 graph.

D. SEARCH SPACE REDUCTION

The known possibility to reduce the search space of cyclic liftings is normalization of the shift values $s_{i,j}$ in (5) [35]. This concept is based on observation that circulating rows of any macro-row $\mathbf{P}^{s_{i,1}}, \mathbf{P}^{s_{i,2}}, \cdots, \mathbf{P}^{s_{i,J}}$ in (5), which affects only the order of parity checks, can give the first $s_{i,j} \neq \infty$ (or last, or any other) in every macro-row normalized to $s_{i,j} = 0$. Similarly, circulating columns of any macro-column, which affects only the order of symbols in a codeword, can give the first $s_{i,j} \neq \infty$ in every macro-column normalized to $s_{i,j} = 0$. Such a row and column reordering within macro-row and macro-column of $\mathbf{H}$ results in a code termed as an equivalent code [35].

Every QC-LDPC code has an equivalent code with normalized parity check matrix, which is defined as a block circulant matrix $\mathbf{H}$ with the first nonzero submatrix in each macro-column and each macro-row equal to $\mathbf{P}^0$. Therefore, the QC-LDPC code search can be restricted to such normalized instances, without the risk of overlooking any well performing code.

Constraining search to only one of the equivalent codes gives a significant reduction of the search space. The general problem formulation remains as in (18)-(19), but since the elements of $\mathbf{s}$ corresponding to submatrices normalized to $\mathbf{P}^0$ are fixed to 0, they are simply removed from $\mathbf{s}$ in (18)-(19), reducing the number of optimized variables. Likewise, the corresponding columns in $\mathbf{A}$ are removed. All numerical results presented in this paper have been obtained for such a normalized matrix search.

Another complexity reduction can be based on an observation that the coefficient vector $\mathbf{a}(d)$ contains equal number of positive and negative elements. Therefore, the majority of closed walks should have the corresponding sum $\mathbf{a}(d) \cdot \mathbf{s}^T$ in the range between $-P$ and P, with the mean value being zero, if there is no a priori knowledge about values in $\mathbf{s}$.

Specifically, we empirically observed that for coefficients $0 \leq s_{i,j} < P$ generated from a uniform random distribution, typically 10-25% of sums $\mathbf{a}(d) \cdot \mathbf{s}^T$ corresponding to cycles of length up to 8, are in the range between $-P$ and P.

If the optimization outcome for eliminated cycles is restricted to this range, the search complexity is reduced, at the cost that some of the non-equivalent solutions cannot be found by an optimization procedure. In such constrained search, only the first and the second alternatives in (12) are considered, thus a simplified search is defined by 5 inequalities (like the first four and the last inequality in (13)), regardless of the cycle length. The component coefficient matrices formulating the simplified optimization problem are presented in (24)-(25). Only 2 additional variables y_u are required for every considered closed walk. We will refer to this method as a reduced search.

$$\mathbf{A}(d) = \begin{bmatrix} \mathbf{a}(d) & 0 & & 0 & -Z & 0 & 0 & & 0 \\ -\mathbf{a}(d) & 0 & & 0 & -Z & 0 & 0 & & 0 \\ -\mathbf{a}(d) & 0 & \cdots & 0 & 0 & -Z & 0 & \cdots & 0 \\ \mathbf{a}(d) & 0 & & 0 & 0 & -Z & 0 & & 0 \\ \mathbf{0} & 0 & & 0 & 1 & 1 & 0 & & 0 \end{bmatrix} \tag{24}$$

$$\mathbf{b}(d)^T = \begin{bmatrix} -1 \\ P-1 \\ -1 \\ P-1 \\ 1 \end{bmatrix} \tag{25}$$

E. PROPOSED CODE DESIGN STRATEGIES

The presented optimization framework can be utilized in a variety of ways, which differ by the length limit $2l_{\max}$ of the examined closed walks, the way of distributing closed walks between strict constraints and optimization goal, as well as an utilization of optional $\alpha(d)$ weights.

Based on the knowledge on structural graph properties, we have investigated a few design strategies, among which the girth conditioning is the basic classical method, while more advanced methods aim at minimizing the number of short cycles with low external connectivity.

1) GIRTH CONDITIONING

One of the commonly applied LDPC code design methods is a search for girth-g graphs. With the presented framework, formulation of a girth-g graph search is straightforward: in the base graph, all closed walks of length up to $(g-2)$ need to be found, for which the strict constraints are formulated by matrices like in (23), while the objective function coefficients $\mathbf{c} = \mathbf{0}$. Then the IP search could succeed giving a girth-g code or fail, which means the girth-g code cannot be obtained for a given base graph $\mathcal{G}$ and lifting degree P.

2) SHORT CYCLES REDUCTION

In a more advanced search than a simple girth conditioning, additionally the existence of girth-length cycles can be reduced by utilizing the objective function. In this method, all closed walks of length up to $2l_{\max} = g$ are found in the base graph, among which the closed walks of length up to $(g-2)$ are reflected as a strict constraints, while length g closed walks are reflected in the objective function with weights $\alpha(d) = 1$. With this formulation, a graph is found characterized by girth-g and minimized number of length g cycles, resulting in a potentially improved code.

3) LOW EXTERNAL CONNECTIVITY CYCLES REDUCTION

The proposed objective function enables incorporating any measure of cycles connectivity into the optimization process, by utilizing the $\alpha(d)$ factor in (21), which can be specified independently for every considered closed walk in the base graph.

We propose a following formulation. All closed walks of length up to $2l_{\max} = g$ should be found in the base graph, among which the closed walks of length up to $(g-2)$ are reflected as a strict constraints. Length-g closed walks are reflected in the objective function with weight $\alpha(d)$ dependent on EMD or ACE (Approximate Cycle EMD) metric [28] of the closed walk. Although it is hard to precisely quantify, how the EMD or ACE is related with the expected harmfulness of the cycle, probably an expression for $\alpha(d)$ relation to ACE can be proposed empirically. For example, we have attempted a design, in which every additional extrinsic message (approximated by ACE) diminish $\alpha(d)$ by a factor of 2, that is $\alpha(d) = 2^{-\mathrm{ACE}(d)}$, where ACE($d$) is the ACE metric of dth closed walk.

4) REDUCED COMPLEXITY SEARCH

A reduced complexity search, as proposed in the previous section, can be utilized with any of the above strategies.

Tab. 1 summarizes the methods we have used for experimental designs, for which numerical results will be presented in section IV.

TABLE 1. Proposed optimization strategies.

Method	
1	Elimination of length up to $(g-2)$ cycles.
2	Elimination of length up to $(g-2)$ cycles; minimization of length-g cycles
3	Reduced search with elimination of length up to $(g-2)$ cycles; minimization of length-g cycles
4	Elimination of length up to $(g-2)$ cycles; minimization of length-g cycles with weight assigned to dth cycle: $\alpha(d) = 2^{-\mathrm{ACE}(d)}$

F. NONBINARY CODES DESIGN

The proposed optimization procedure can be employed as an initial step for a nonbinary code construction. Given a binary parity check matrix $\mathbf{H}$ as in (5), a nonbinary structured block-circulant matrix over GF(q) can be obtained by substituting nonzero entries in $\mathbf{H}$ with elements from GF(q). While some research papers apply randomly generated entries, an optimized selection can give better results.

We have employed a design that is inspired by a thorough discussion found in [44], where the binary images method

for selection on a row of $\mathbf{H}$ basis was proposed. Since in a cyclically lifted graph of structured codes, the topological structures in the graph are P times duplicated, fixing the nonzero entries for every circulant of $\mathbf{H}$ allows application of the ideas from [44], while reducing the number of coefficients that need to be memorized. Therefore in our nonbinary construction, every nonzero circulant $\mathbf{P}^{s_{d,l}}$ in (5) is wholly multiplied by an element $g_{d,l} \in \mathrm{GF}(q)$. Hence, the nonzero entries in $\mathbf{H}$ (coefficients) are selected on a base matrix level.

For every row of the base matrix $\mathbf{W}$, an initial independent choice of a set of coefficients is made, subject to the row degree. The coefficient set can be chosen from a precomputed collection of sets presented in [44], but results provided in that research cover only rows of degree $d_c = 4$. Therefore, with the implementation of a similar search method, we have computed the collections of optimized coefficients sets for rows of the base matrix. Collections of two candidate sets for GF(8), GF(16) and GF(64) nonzero entries in rows of degree up to 6 are presented in Tab. 2. The Galois Field elements are presented in the power notation α^i of the primitive element α.

TABLE 2. Precomputed row coefficients for GF(8), GF(16), GF(64) and different row degrees d_c [47]; the numbers represent the exponent i in the power representation α^i for elements of GF(q).

d_c	GF(8)	GF(16)	GF(64)
3	{0, 1, 4} {0, 2, 4}	{0, 4, 8} {0, 5, 10}	{0, 15, 41} {0, 16, 41}
4	{0, 1, 3, 4} {0, 1, 3, 5}	{0, 3, 7, 11} {0, 2, 6, 11}	{0, 9, 22, 37} {0, 7, 18, 44}
5	{0, 1, 2, 3, 5} {0, 1, 2, 4, 5}	{0, 1, 4, 8, 11} {0, 1, 5, 7, 11}	{0, 7, 18, 44, 53} {0, 7, 33, 42, 55}
6	{0, 1, 2, 3, 4, 5}	{0, 1, 4, 6, 8, 11} {0, 1, 4, 8, 11, 12}	{0, 7, 33, 42, 49, 55} {0, 6, 13, 19, 43, 52}

Every row of the base matrix initially comprises one of the precomputed sets of coefficients arranged in an arbitrary order. Then additional conditioning method is incorporated: a greedy search loop iterates over all cycles that exist after lifting process, and tries to modify: 1) the coefficient set choices, 2) the orders of the coefficients in respective rows and 3) the multiplicative factor, in order to make as many cycles as possible irresolvable [44], [48].

G. OVERALL QC-LDPC CONSTRUCTION ALGORITHM

In order to summarize the discussion, we provide a concise description of the whole proposed QC-LDPC code construction method, presented below as an Algorithm 1. The algorithm can be used to design (N, K) QC-LDPC codes with $N = JP$, $K = (J - I)P$ and $\mathbf{H}$ structure as in (5). Regular or irregular codes can be designed, with column degree distribution defined by $\boldsymbol{\lambda} = [\lambda_2, \lambda_3, \ldots, \lambda_{\max(d_v)}]$, where λ_i specifies a share of degree-i columns in the base matrix $\mathbf{W}$ (and consequently in $\mathbf{H}$). In the case of irregular codes, the distribution should be pre-optimized for capacity approaching with existing methods, such as Density Evolution or EXIT charts.

Construction of the base graph is supported by the classic Progressive Edge Growth (PEG) algorithm [49], [50], which by greedy optimization minimizes existence of the shortest cycles already on the base graph construction level. Any other base graph construction can be easily utilized too. If a dual-diagonal feature is required, the base matrix should be constrained to possess the dual-diagonal structure.

Algorithm 1 QC-LDPC Code Over GF(q) Construction

Input: Base matrix size $(I \times J)$, submatrix size P, column degree distribution $\boldsymbol{\lambda}$, closed walks length detection limit g, GF order q.

Output: QC-LDPC parity check matrix $\mathbf{H}_{IP \times JP}$

1. Construct binary base matrix $\mathbf{W}_{I \times J}$ subject to the column degree distribution $\boldsymbol{\lambda}$, utilizing the PEG method [49] with concentrated row distribution [50].
2. Analyze the base graph associated with $\mathbf{W}$: identify all closed walks of length $2l < g$ (method 1) or $2l \leq g$ (methods 2–4).
3. For $d = 1, \ldots, D$, where D is the number of identified closed walks, create IP constraint coefficient submatrices $\mathbf{A}(d)$ and $\mathbf{b}(d)^T$ according to (23) or (24)-(25).
4. For $d = 1, \ldots, D$, create goal function coefficients $\mathbf{c}(d)$ according to (21), where:
   – for method 1, $\alpha(d) = 0$;
   – for methods 2 and 3, $\alpha(d) = \begin{cases} 1, & L(d) = g \\ 0, & L(d) < g \end{cases}$;
   – for method 4, $\alpha(d) = \begin{cases} 2^{-\mathrm{ACE}(d)}, & L(d) = g \\ 0, & L(d) < g \end{cases}$.
5. With $\mathbf{A} = \mathrm{col}[\mathbf{A}(1), \mathbf{A}(2), \cdots, \mathbf{A}(D)]$, $\mathbf{b} = \mathrm{col}[\mathbf{b}(1), \mathbf{b}(2), \cdots, \mathbf{b}(D)]$ and $\mathbf{c} = [\mathbf{c}(1), \mathbf{c}(2), \cdots, \mathbf{c}(D)]$, solve the IP problem defined by expressions (18)-(19).
6. With solution $\mathbf{s} = [s_1, s_2, \ldots, s_T]$ construct $\mathbf{H}$ as in (5).
7. **if** $q > 2$ **then**
8. For every row in $\mathbf{W}$, randomly select a set of nonzero coefficients over GF(q) from a collection of sets as in Tab. 2, for a given row degree.
9. Optionally: iteratively modify the coefficients as outlined in section III-F.
10. Multiply every submatrix $\mathbf{P}^{s_i}$ in $\mathbf{H}$ by a selected GF(q) coefficient.

The next step, detecting all closed walks of length $2l$ up to g in the created base graph, is performed with an algorithm similar to the search method utilizing the support tree [28], also known as cycle generating tree [35]. The search algorithm is based on a remark that each closed walk of length $2l$ can be considered as a concatenation of two paths of length l. For every variable node v_j in $\{v_1, v_2, \ldots, v_J\}$, in order to find all closed walks of length up to g passing v_j, a support tree of all paths from v_j of length up to $g/2$ is created. A closed walk of length-$2l$ is marked if two positions at level-l of support tree represent the same node in the base graph. In order to avoid multiple detection of the same closed walk, the tree expansion for v_j does not include nodes $v_{j'}$ for $j' < j$.

In principle, the number of nodes in the support tree grows exponentially with the number of levels expanded. However, the tree storage complexity can be greatly reduced using a compressed data structure, which corresponds to a trellis-like representation of support tree [28]. In an implementation oriented approach that we applied, at every expansion level l, the algorithm memorizes: 1) the subset of nodes $\mathcal{V}(l)$ that is reached at this level, which is $\mathcal{V}(l) \subseteq \mathcal{V}_c$ at odd levels or $\mathcal{V}(l) \subseteq \mathcal{V}_v$ at even levels; 2) for every node in $\mathcal{V}(l)$: list of all edges (v_j, c_i) that connects this node with level-$(l-1)$ nodes. In such an approach, at each level at most I (for odd l) or J (for even l) nodes and at most $I \times \bar{d}_c = J \times \bar{d}_v$ edges have to be memorized, where $\bar{d}_c$ and $\bar{d}_v$ are average row and column degrees respectively. The total number of memorized edges for levels up to $g/2$ is then $I \times \bar{d}_c \times g/2$, which means it is linearly dependent on the search depth g.

After creating the trellis like structure, for every node at level l with multiple connections to the previous level, the trellis has to be backtracked, marking all the possible paths as different closed walks existing in the base graph. The number of closed walks is still exponentially dependent on g, however complexity of such a search method can be easily limited by introducing a numerical limit $D_{\max}$ on the number of closed walks that should be detected. The line 2 of Algorithm 1 can be then reformulated for example as follows:

Analyze the base graph associated with $\mathbf{W}$:
For every v_j in $\{v_1, v_2, \ldots, v_J\}$ create a trellis reflecting the support tree, excluding $v_{j'}$ for $j' < j$.
For subsequent levels $l = 2, 3, \ldots$, backtrack the trellis and list the closed walks. If for any l the number of detected closed walks exceeds $D_{\max}$, break the loop and assign $g := 2l$.

This way, the Algorithm 1 can proceed with $D_{\max}$ as an input parameter instead of girth g and complexity of the base graph analysis is linearly dependent on $D_{\max}$. Meanwhile, we have observed empirically that typically (for small to moderate sizes P) setting $D_{\max}$ to be more than a few thousands is aimless, because extending the list of closed walks over this length does not further improve final results of the optimization.

After closed walks detection, the IP problem is formulated and solved, as indicated in Algorithm 1 and described in the previous sections. Finally, if a nonbinary code is desired (over GF(q) with $q > 2$), the coefficients over GF(q) should be generated, for every nonzero element in the base matrix $\mathbf{W}$, corresponding to every submatrix $\mathbf{P}^{s_i}$ in $\mathbf{H}$.

IV. NUMERICAL RESULTS

Experimental studies have been based on Monte Carlo simulations, evaluating error correction performance of the constructed structured LDPC codes, binary as well as nonbinary cases. Numerical results provided in this section show effectiveness of the proposed method. Codes of block length up to a few thousands have been successfully created with the IP optimization. The utilized simulation model involves BPSK modulation, AWGN channel and log-domain decoding algorithms: LLR-BP for binary codes and FFT-QSPA [51] belief propagation for nonbinary codes, with the maximum number of iterations set to 100.

A. BINARY QC-LDPC CODES

Binary QC-LDPC codes with a broad range of lengths, rates and submatrix sizes have been constructed. Here we present a portion of the obtained results.

Block Error Rate (BLER) performances of the rate-1/2 codes of length $N = 576$ and $N = 1004$ are shown in Fig. 3. The designed irregular (576,288) codes have the same degree distribution as a code defined for WiMAX. However, since the WiMAX codes are known to be easily outperformed in the error floor region, performance of a recent (576,288) QC-LDPC code [41], obtained by array dispersion with masking, is also used as a reference. In our design, a base graph corresponding to a matrix similar to $\mathbf{M}_2$ in [41] has been utilized. The cyclic liftings have been optimized with methods, which we refer to as method 2 and 4 in Tab. 1. Both designed codes greatly outperform the WiMAX code and they also perform better than the recent QC-LDPC code [41], achieving an additional coding gain of about 0.2dB in the error floor region.

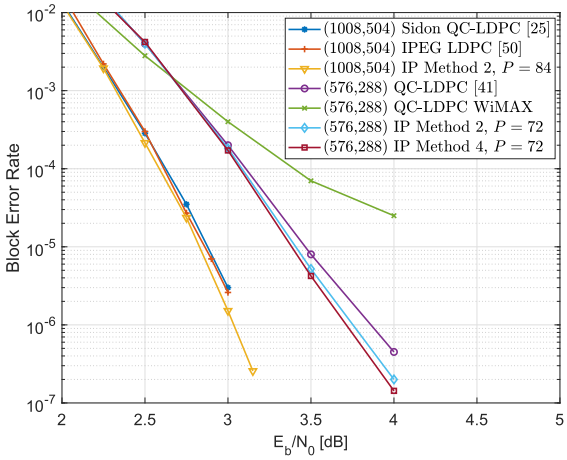

FIGURE 3. Performance of designed rate-1/2 binary codes in comparison with QC-LDPC codes [41], [25] and IPEG codes [50].

In the same Figure, a regular (1008,504) code obtained with Method 2 is compared with a regular algebraically constructed QC-LDPC code based on Sidon sequence [25] as well as a non-structured modified PEG (Progressive Edge Growth) code [50]. A slight performance gain can be observed for the IP designed code. The performance gain increases for larger SNRs and for the designed code no error floor is observed down to the BLER of nearly 10^{-7}.

Performances of higher rate binary codes are shown in Fig. 4. An IP designed (576,432) code is compared with WiMAX code and a recent QC-LDPC code [41], both codes with the same degree distribution. Again, a performance gain over QC-LDPC code from [41] can be observed, which grows

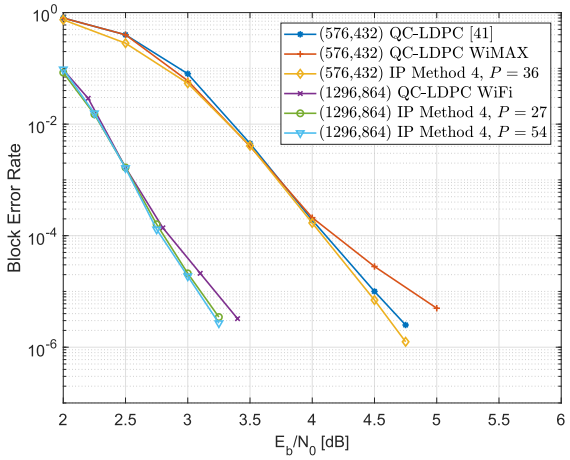

FIGURE 4. Performance of designed rate-2/3 and rate-3/4 binary codes in comparison with QC-LDPC codes [41] and WiMAX codes.

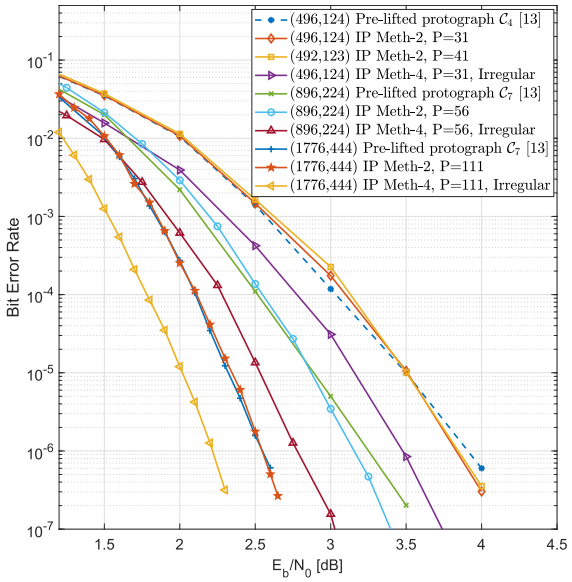

FIGURE 5. Performance of designed rate-1/4 binary codes in comparison with pre-lifted protograph based codes [13].

in the error floor region. We have also designed irregular (1296,864) codes, with degree distribution taken from a WiFi 802.11n code, preserving the dual-diagonal structure of WiFi parity check matrix. Results for codes with two different submatrix sizes: $P = 54$ and $P = 27$ are presented in Fig. 4. The code with $P = 54$ is constructed with the same base matrix as a WiFi code and the code with $P = 27$ is designed from a dual-diagonal base matrix constructed utilizing the PEG algorithm. We can see a significant performance gain in the error floor region over WiFi code for both submatrix sizes. In general, for all the designed binary codes, the demonstrated performance gain is increasing in the error floor region.

In Fig. IV-A we present a Bit Error Rate (BER) comparison of the designed QC-LDPC codes with rate-1/4 codes based on pre-lifted protographs [13]. Best performing codes $\mathcal{C}_4$ and $\mathcal{C}_7$ of lengths $N = 496$, $N = 896$ and $N = 1776$ have been taken for comparison from [13]. With the proposed IP optimization, codes with the same lifting degree ($P = 31$, $P = 56$ and $P = 111$ respectively) and the same regular $d_v = 3$ distribution have been constructed and simulated. Additionally, a code with smaller base matrix and $P = 41$ have been also constructed with the same $N = 496$ length. All the designed codes have less short cycles in their Tanner graphs than the reference codes. For example:

- the length-496 code $\mathcal{C}_4$ has a girth-8 graph with 186 8-cycles and 961 10-cycles,
- while the length-496 code designed in this work has a girth-10 graph with no 8-cycles and 465 10-cycles.

Also the length-1776 code designed in this work has greater girth – 12 than similar length-1776 code from [13], which is characterized by girth 10. However, the BER performance results for these two codes are very similar (Fig. IV-A), perhaps because the performance curve in [13] ends before any error floor is observed. Meanwhile, for the length-496 and length-896 codes, performance improvement of the proposed codes is clearly visible in the provided results. Moreover, with IP method-4 we have designed codes with the same lengths, but characterized by the following irregular degree distribution: $\boldsymbol{\lambda} = [\lambda_2 = 0.4375, \lambda_3 = 0.375, \lambda_4 = 0.0626, \lambda_6 = 0.125]$. The simulation results, also presented in Fig. IV-A, show that the designed irregular codes outperform their regular counterparts by about 0.4 dB, while the error floor is not detected down to BER of about 10^{-7}.

B. CODES OVER GF(8)

The semi-regular 6×12 base matrix with mean column degree 2.5 presented in [38, eq. (12)], has been used as a basis of a thorough comparative studies. Huang *et al.* [38] report the smallest sizes of submatrix P (denoted by L in [38]), for which their designed computer search technique enables to achieve the assumed girth.

Our study revealed that the base graph corresponding to the 6×12 base matrix contains: 9 different length-4 closed walks, 56 length-6 closed walks, 209 length-8 closed walks, 1028 length-10 closed walks. The proposed optimization with an IP numerical solver successfully achieved codes of the same g and P parameters as in [38]. Moreover, some attempts to construct code with the same girth, but smaller size P were also successful. In Tab. 3 we report the smallest sizes of submatrix P, for which girth up to 10 was achievable in our study altogether with results from [38]. We provide some of the obtained parity check matrices in Fig. 6.

TABLE 3. Minimum submatrix sizes achieved for girth up to 10 codes from base matrix (12) in [38].

Girth	[38]	$L_{\min}$, [38]	$P_{\min}$ this work
6	Design 1	4	3
8	Design 2	8	6
10	Design 3	24	23

	1	2	3	4	5	6	7	8	9	10	11	12
1	0		0		0		0	0				
2		0		0		0	2	0				
3	0		4			3			0	0		
4		2		4	3				4	0		
5	4			3	0						0	0
6		2	5			0					2	0

(a)

	1	2	3	4	5	6	7	8	9	10	11	12
1	0		0		0		0	0				
2		0		0		0	22	0				
3	13		20			4			0	0		
4		12		0	15				1	0		
5	15			5	12						0	0
6		2	16			4					6	0

(b)

FIGURE 6. Permutation shift values obtained by the IP optimization that from the base matrix (12) in [38] give: (a) girth-8 code with $P = 6$, (b) girth-10 code with $P = 23$. Empty squares correspond to all-zero submatrices in H.

Then, utilizing the same base matrix, codes over GF(8) have been constructed with symbol lengths 228 ($P = 19$), 492 ($P = 41$) and 756 ($P = 63$). Method 2 has been applied with $g = 6$, $g = 8$ and $g = 10$ respectively, and the nonzero coefficients over GF(8) have been obtained as outlined in section III-F. Fig. 7 shows performance of the constructed codes in comparison with codes originating from the same base matrix designed in [38]. A slight performance gain can be observed for proposed IP optimization over heuristic search from [38].

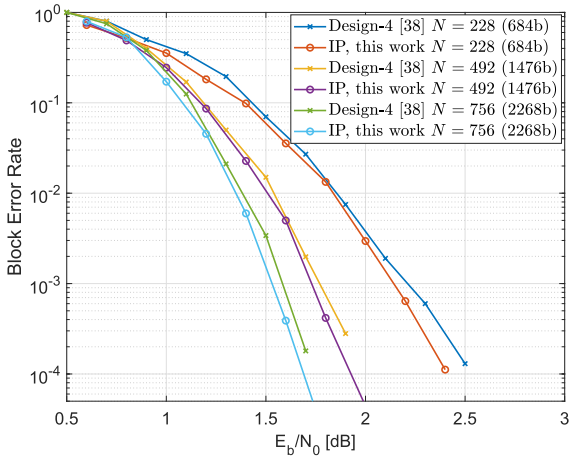

FIGURE 7. Performance of designed codes over GF(8) with various lengths constructed from base matrix (12) in [38].

C. CODES OVER GF(64)

Codes of length 1008b over GF(64) have been investigated in relation to girth optimized regular cycle codes with optimized nonzero entries referred to as 'Opt-2' in [39]. Classic PEG (Progressive Edge Growth) algorithm has been used to construct the base matrices of sizes: 12×24 that has been lifted to obtain code (168,84) with submatrix size $P = 7$, as well as 4×8 that after lifting gives also code (168,84), but with submatrix size $P = 21$. Regular ($d_v = 2$,$d_c = 4$) base matrices lead to cycle codes. However, we have also designed semi-regular codes with mean variable node degree $\bar{d}_v > 2$, among which the best performance was observed for $\bar{d}_v = 2.083$, obtained from 12×24 base matrix with 2 degree-3 columns.

We have employed methods 1-4 (Tab. 1) with $g = 12$ for cycle codes and $g = 10$ for semi-regular codes, since due to the denser matrix, girth-12 was not achievable in this case. For cycle codes, method 4 was not applied, because it gives the same result as method 2, since ACE metrics are the same for all cycles. Fig. 8 shows performance of the constructed codes with different methods, while Fig. 9 shows comparison with PEG cycle code, GF(64) cycle code from [39] (Fig. 4 therein) as well as binary counterpart 1008b, rate-1/2 QC-LDPC code based on Sidon sequence [25].

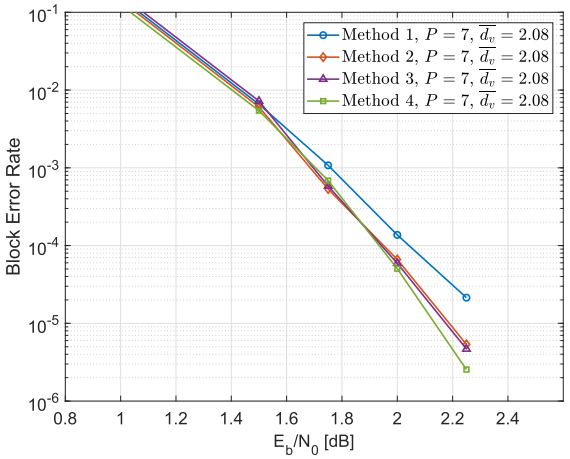

FIGURE 8. Performance of designed codes of length $N = 168$ over GF(64) constructed by different proposed methods from a base matrix obtained by PEG algorithm.

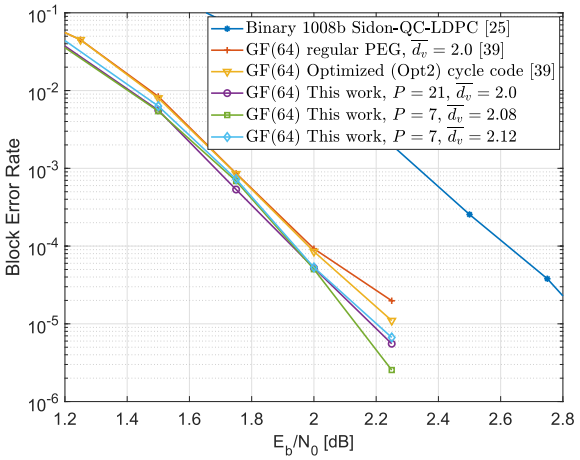

FIGURE 9. Performance of designed codes of length $N = 168$ over GF(64) constructed from a base matrix obtained by PEG algorithm, and their counterparts from [25] and [39].

It can be noticed that performance of girth conditioned code (as in Method 1) can be significantly improved by simultaneous elimination of girth-length cycles, like in methods 2-4. Meanwhile, codes obtained with methods 2-4 perform quite similarly.

Fig. 9 shows that the constructed codes obviously outperform the binary counterpart, and also outperform PEG code over GF(64) as well as the GF(64) counterpart from [39],

particularly in the error floor region. It can be also observed that among the designed codes with different node degree distribution, semi-regular code with $\bar{d}_v = 2.083$ is the best choice, slightly outperforming regular cycle code.

D. CODE OVER GF(256)

Constant degree $d_v = 2$ leading to a cycle code is known to be an optimal choice for degree distribution for codes over high-order GF fields, particularly GF(256). The cycle code that has been designed has block-length of 896b, that is 112 symbols over GF(256), and it is a direct counterpart to the code with parity check matrix provided in (40)-(41) in [35]. The code has been designed with method 2 (Tab. 1) with $g = 12$, utilizing the base matrix extracted from (40) in [35] and coefficients over GF(256) extracted from (41) in [35]. Then, the only difference is in the permutation shift values search, which is the proposed IP-based method in our design and a greedy pre-search with final selection by simulation of candidate codes in design [35]. We have also constructed a similar code, but with optimized coefficients as presented in section III-F, instead of semi-random choice of [35].

Numerical results are presented in Fig. 10. In order to observe any difference in performance of the simulated codes, it was necessary to extend the simulation beyond SNR 2 dB, which is a limit of results provided in [35]. Therefore, the presented performances result from our simulations, also for the code (40)-(41) in [35]. All 3 codes perform similarly in the waterfall region. However, a performance improvement of the designed codes can be observed in the error floor region, which is in the order of 0.1 dB at the 10^{-7} BER level. The error floor level for the designed codes is significantly lower than for the reference code.

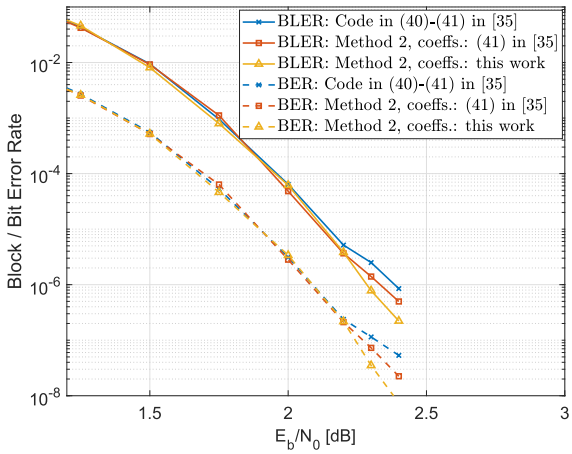

FIGURE 10. Performance of codes over GF(256) constructed from an 8 × 16 base matrix taken from [35]; Block and Bit Error Rates are illustrated with solid and dashed lines respectively.

V. CONCLUSIONS

It has been presented that the protograph lifting for flexible QC-LDPC code construction can be formulated as an optimization problem with linear constraints and linear objective function for cycles elimination in the code graph. Then, a Mixed Integer Linear Programming (MILP) application can solve the QC-LDPC code optimization problem. The proposed optimization goal function, which is global, that is it reflects all short closed walks in the base graph up to the length limit $D_{\max}$, is very flexible and enables weighting by metrics characterizing cycles. Binary and nonbinary QC-LDPC codes, regular and irregular, have been shown to be effectively constructed with the proposed optimization method.

Presented design case studies for binary and nonbinary codes reveal effectiveness of the proposed design strategies for base graph lifting. Since the IP solver could reach a global optimum of the objective function, the constructed codes often outperformed their counterparts from previous research. For all the investigated design cases, a significant performance gain is demonstrated mainly in the error floor region. It is not surprising, because the graph optimization is known to influence primarily the error floor performance. Meanwhile, irregular designs with optimized degree distribution achieved a significant waterfall region coding gains without error floor at least down to the BLER of about 10^{-7}.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.
[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[3] M. C. Davey and D. MacKay, "Low-density parity check codes over $GF(q)$," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.
[4] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for designing LDPC codes using protographs and circulants," in *Proc. Int. Symp. Inf. Theory*, Chicago, IL, USA, Jun. 2004, pp. 236–238.
[5] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.
[6] A. K. Pradhan, A. Thangaraj, and A. Subramanian, "Construction of near-capacity protograph LDPC code sequences with block-error thresholds," *IEEE Trans. Commun.*, vol. 64, no. 1, pp. 27–37, Jan. 2016.
[7] Y. Fang, G. Bi, and Y. L. Guan, "Design and analysis of root-protograph LDPC codes for non-ergodic block-fading channels," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 738–749, Feb. 2015.
[8] Y. Fang, Y. L. Guan, G. Bi, L. Wang, and F. C. M. Lau, "Rate-compatible root-protograph LDPC codes for quasi-static fading relay channels," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2741–2747, Apr. 2016.
[9] Y. Fang, S. C. Liew, and T. Wang, "Design of distributed protograph LDPC codes for multi-relay coded-cooperative networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7235–7251, Nov. 2017.
[10] F. Steiner, G. Böcherer, and G. Liva, "Protograph-based LDPC code design for shaped bit-metric decoding," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 397–407, Feb. 2016.
[11] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.
[12] M. Stinner and P. M. Olmos, "On the waterfall performance of finite-length SC-LDPC codes constructed from protographs," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 345–361, Feb. 2016.
[13] D. G. M. Mitchell, R. Smarandache, and D. J. Costello, Jr., "Quasi-cyclic LDPC codes based on pre-lifted protographs," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5856–5874, Oct. 2014.
[14] Y. Fang, G. Bi, Y. L. Guan, and F. C. M. Lau, "A survey on protograph LDPC codes and their applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1989–2016, 4th Quart., 2015.
[15] L. Dolecek, D. Divsalar, Y. Sun, and B. Amiri, "Non-binary protograph-based LDPC codes: Enumerators, analysis, and designs," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3913–3941, Jul. 2014.

[16] R. Wang, Y. Li, and H. Zhao, "Nonbinary protograph-based LDPC codes based on additive group of finite field," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 636–639, Apr. 2016.
[17] W. Sułek, "Quasi-regular QC-LDPC codes derived from cycle codes," *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1705–1708, Sep. 2016.
[18] S. Zhao and X. Ma, "Construction of high-performance array-based non-binary LDPC codes with moderate rates," *IEEE Commun. Lett.*, vol. 20, no. 1, pp. 13–16, Jan. 2016.
[19] C. Chen, B. Bai, G. Shi, X. Wang, and X. Jiao, "Nonbinary LDPC codes on cages: Structural property and code optimization," *IEEE Trans. Commun.*, vol. 63, no. 2, pp. 364–375, Feb. 2015.
[20] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
[21] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic LDPC codes for fast encoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 8, pp. 2894–2901, Aug. 2005.
[22] W. Sułek, "Non-binary LDPC decoders design for maximizing throughput of an FPGA implementation," *Circuits, Syst., Signal Process.*, vol. 35, pp. 4060–4080, Nov. 2016.
[23] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2429–2458, Jul. 2007.
[24] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1652–1662, Jun. 2009.
[25] G. Zhang, R. Sun, and X. Wang, "New quasi-cyclic LDPC codes with girth at least eight based on Sidon sequences," in *Proc. 7th Int. Symp. Turbo Codes Iterative Inf. Process.*, Gothenburg, Sweden, Aug. 2012, pp. 31–35.
[26] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, "Algebraic quasi-cyclic LDPC codes: Construction, low error-floor, large girth and a reduced-complexity decoding scheme," *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 2626–2637, Aug. 2014.
[27] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, VA, USA, Oct. 2003, pp. 1426–1435.
[28] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.
[29] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Lowering the error floor of LDPC codes using cyclic liftings," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2213–2224, Apr. 2011.
[30] X. Zheng, F. C. M. Lau, and T. K. Chi, "Constructing short-length irregular LDPC codes with low error floor," *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2823–2834, Oct. 2010.
[31] G. Han, Y. L. Guan, and L. Kong, "Construction of irregular QC-LDPC codes via masking with ACE optimization," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 348–351, Feb. 2014.
[32] A. Tasdighi, A. H. Banihashemi, and M. R. Sadeghi, "Efficient search of girth-optimal QC-LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1552–1564, Apr. 2016.
[33] H. Xu and B. Bai, "Superposition construction of Q-ary LDPC codes by jointly optimizing girth and number of shortest cycles," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1285–1288, Jul. 2016.
[34] Y. Wang, S. C. Draper, and J. S. Yedidia, "Hierarchical and high-girth QC LDPC codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4553–4583, Jul. 2013.
[35] I. E. Bocharova, B. D. Kudryashov, and R. Johannesson, "Searching for binary and nonbinary block and convolutional LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 163–183, Jan. 2016.
[36] F. Amirzade and M.-R. Sadeghi, "Lower bounds on the lifting degree of QC-LDPC codes by difference matrices," *IEEE Access*, vol. 6, pp. 23688–23700, Apr. 2018.
[37] L. Yang, H. Liu, and C.-J. R. Shi, "Code construction and FPGA implementation of a low-error-floor multi-rate low-density parity-check code decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 4, pp. 892–903, Apr. 2006.
[38] J. Huang, L. Liu, W. Zhou, and S. Zhou, "Large-girth nonbinary QC-LDPC codes of various lengths," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3436–3447, Dec. 2010.
[39] J. Huang, S. Zhou, and P. Willett, "Structure, property, and design of nonbinary regular cycle codes," *IEEE Trans. Commun.*, vol. 58, no. 4, pp. 1060–1071, Apr. 2010.
[40] A. Bajpai, A. Kalsi, S. Nakpeerayuth, P. Kovintavewat, and L. Wuttisittikulkij, "A greedy search based method with optimized lower bound for QC-LDPC codes," in *Proc. IEEE 13th Int. Symp. Auto. Decentralized Syst.*, Bangkok, Thailand, Mar. 2017, pp. 165–168.
[41] H. Xu, D. Feng, R. Luo, and B. Bai, "Construction of quasi-cyclic LDPC codes via masking with successive cycle elimination," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2370–2373, Dec. 2016.
[42] M. Karimi and A. H. Banihashemi, "On the girth of quasi-cyclic protograph LDPC codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4542–4552, Jul. 2013.
[43] A. Tasdighi, A. H. Banihashemi, and M.-R. Sadeghi, "Symmetrical constructions for regular girth-8 QC-LDPC codes," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 14–22, Jan. 2017.
[44] C. Poulliat, M. Fossorier, and D. Declercq, "Design of Regular $(2,d_c)$-LDPC Codes over GF(q) Using Their Binary Images," *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008.
[45] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[46] R. Diestel, *Graph Theory*. Berlin, Germany: Springer-Verlag, 2006.
[47] W. Sulek and M. Kucharczyk, "Partial parallel encoding and algorithmic construction of non-binary structured IRA codes," *China Commun.*, vol. 13, no. 8, pp. 103–116, 2016.
[48] J. Huang and J. Zhu, "Linear time encoding of cycle GF(2^q) codes through graph analysis," *IEEE Commun. Lett.*, vol. 10, no. 5, pp. 369–371, May 2006.
[49] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[50] H. Chen and Z. Cao, "A modified PEG algorithm for construction of LDPC codes with strictly concentrated check-node degree distributions," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Kowloon, China, Mar. 2007, pp. 564–568.
[51] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. IEEE Inf. Theory Workshop*, Paris, France, Mar. 2003, pp. 70–73.

WOJCIECH SUŁEK received the Ph.D. degree in electronics from the Silesian University of Technology, Gliwice, Poland, in 2009. He is currently an Assistant Professor with the Institute of Electronics, Silesian University of Technology. His Ph.D. thesis was regarding Architecture Aware LDPC Codes Construction and Decoder Implementation. His current research interests include modern coding theory and coding systems hardware design. Results of his research in the area of LDPC codes design and implementation have been published in numerous conferences and journals, including the IEEE TRANSACTIONS ON COMMUNUNICATIONS, the IEEE COMMUNICATIONS LETTERS, and the IEEE ACCESS.